
 [Zeecoders](#) / [LeetCode](#)

[Watch](#) 2 [Star](#) 6 [Fork](#) 2

[Code](#) [Issues](#) 0 [Pull requests](#) 0 [Wiki](#) [Pulse](#) [Graphs](#)




Branch: master ▾ [LeetCode](#) / [【一天一道LeetCode】#60. Permutation Sequence..md](#)

[Find file](#) [Copy path](#)

 [ZCheng1004](#) 2016-7-14 by Zeecoder cc793e2 7 days ago

1 contributor

77 lines (76 sloc) | 2.43 KB

[Raw](#) [Blame](#) [History](#)   

# 一天一道LeetCode系列

## (一) 题目

The set [1,2,3,...,n] contains a total of n! unique permutations. By listing and labeling all of the permutations in order, We get the following sequence (ie, for n = 3): 1: "123" 2: "132" 3 : "213" 4 : "231" 5 : "312" 6 : "321" Given n and k, return the kth permutation sequence.

## (二) 解题

### 第一种解法：

参考：[【一天一道LeetCode】#31. Next Permutation](#) 这篇博客，很奇怪的是为什么n=8,k=8590的时候会超时，暂时还没有想出来原因，欢迎大家留言讨论。

```
/*
利用STL的next_permutation每次求出下一个排列数
*/
class Solution {
public:
    string getPermutation(int n, int k) {
        string seq;
        string ret;
        for(int i = 0 ; i < n ; i++)
        {
            seq+=('1' + i);
        }
        do
        {
            k--;
            if(k==0) {
                ret = seq;
                break;
            }
        }while(next_permutation(seq.begin(),seq.end()));
        return ret;
    }
};
```

### 第二种解法：

利用排列数的规律来求解。我们以题目给出的例子为例来讲解规律。n=3时，由1，2，3三个数组成{a1，a2，a3} 1: "123" 2: "132" 3 : "213" 4 : "231" 5 : "312" 6 : "321" 首先看第一个数a1（候选数字temp = "123"），k为1,2时，a1=1，k为3,4时，a1=2，k为5,6时，a1=3，从中可以看出a1=temp[(k-1)/(n-1)!]。确定了第一个数后我们来看第二个数

a2, 以1,2这一组来看, 排除了1, 候选数字temp = "23", 这个时候k只能为1和2, 所以在上一步中k%=(n-1)!, 这个时候a2=temp[(k-1)/(n-2)!] 最后a3只能为剩下的一个数字了。

ok, 整理一下思路, 我们需要一个候选字符串temp, 一个(n-1)!的数组f[10] = {1,1,2,6,24,120,720,5040,5040\*8} (n为1~9的数字), k初始值为k-1 在每一步中, a1=temp[k/f[n-1]], k%=f[n-1]。一直到n为0。具体思路见代码:

```
class Solution {
public:
    string getPermutation(int n, int k) {
        string temp = "123456789";
        int f[] = {1,1,2,6,24,120,720,5040,5040*8};
        string ret;
        int i = n;
        k--; //k的初始值
        while(i)
        {
            int m = k/f[i-1];
            k %= f[i-1];
            ret += temp[m];
            temp.erase(temp.begin()+m); //擦除掉已经选掉的值
            i--;
        }
        return ret;
    }
};
```

